(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **11212456 A**

(43) Date of publication of application: **06.08.99**

(51) Int. Cl **G09C 1/00**
G06F 7/72
H04L 9/30

(21) Application number: **10014681**

(22) Date of filing: **27.01.98**

(71) Applicant: **FUJITSU LTD**

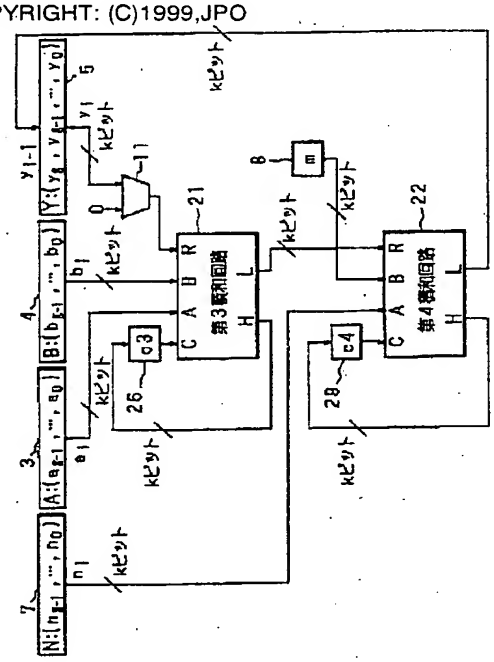
(72) Inventor: **TAKENAKA MASAHIKO**
ITO KOICHI
TORII NAOYA

(54) MULTIPLICATION REMAINDER CALCULATION DEVICE USING MONTGOMERY METHOD

(57) Abstract:

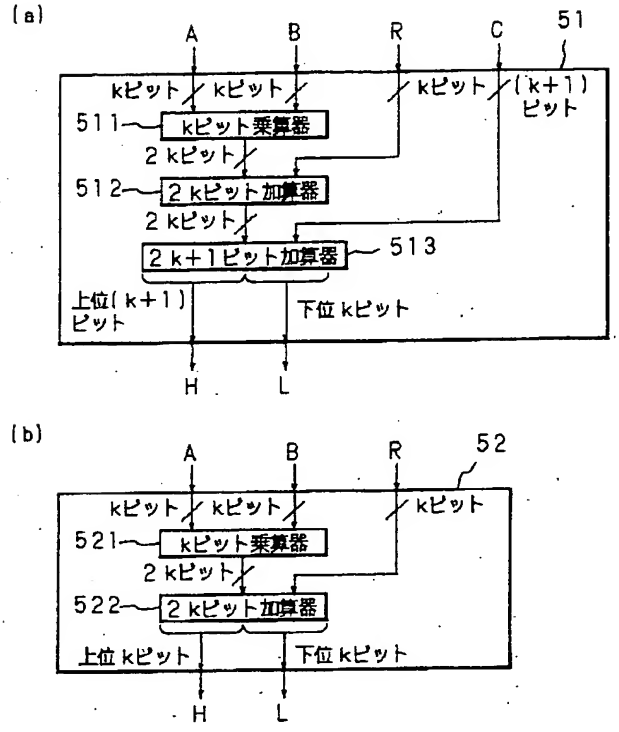
PROBLEM TO BE SOLVED: To provide a calculation device which simplifies the constitution of a product sum circuit and permits pipeline processing and uses a Montgomery algorithm to quickly perform multiplication remainder calculation.

SOLUTION: A product sum circuit 21 multiplies outputs of an A register 3 and a B register 4 and adds outputs of a c3 register 26 and a Y register 5 to the multiplication result. A product sum circuit 22 multiplies outputs of an N register 7 and an m register 8 and adds outputs of a c4 register 29 and the product sum circuit 21 to the multiplication result. Registers 26 and 29 for carry in two product sum circuits 21 and 22 are provided independently of each other, and carry is returned to the corresponding product sum circuit. All the processing is performed in a processing unit (k bits). The next operation of the product sum circuit 21 can be performed during the operation of the product sum circuit 22.



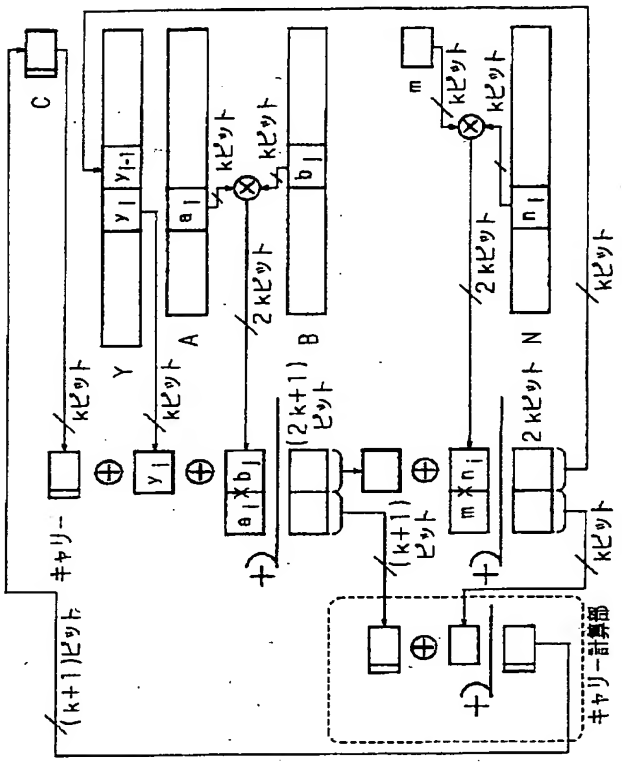
【図１２】

従来の積和回路の構成図



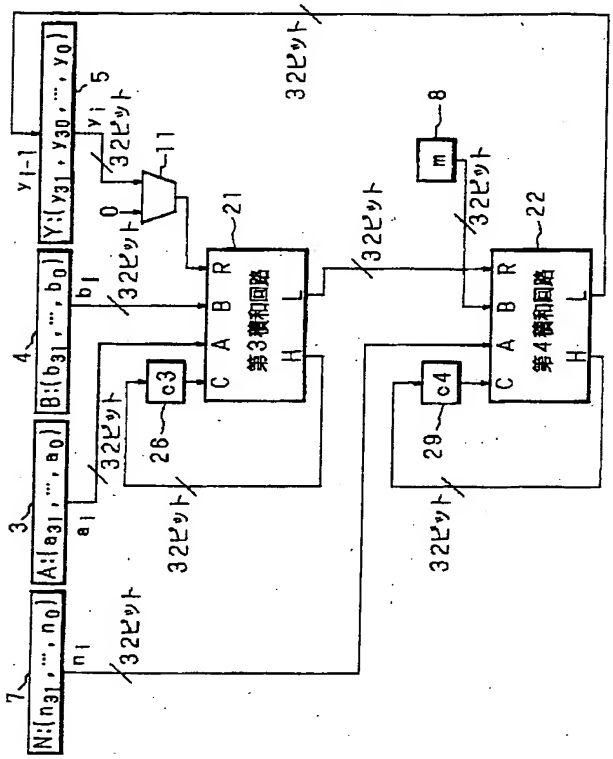
【図１３】

従来の乗算剰余計算装置の動作説明図



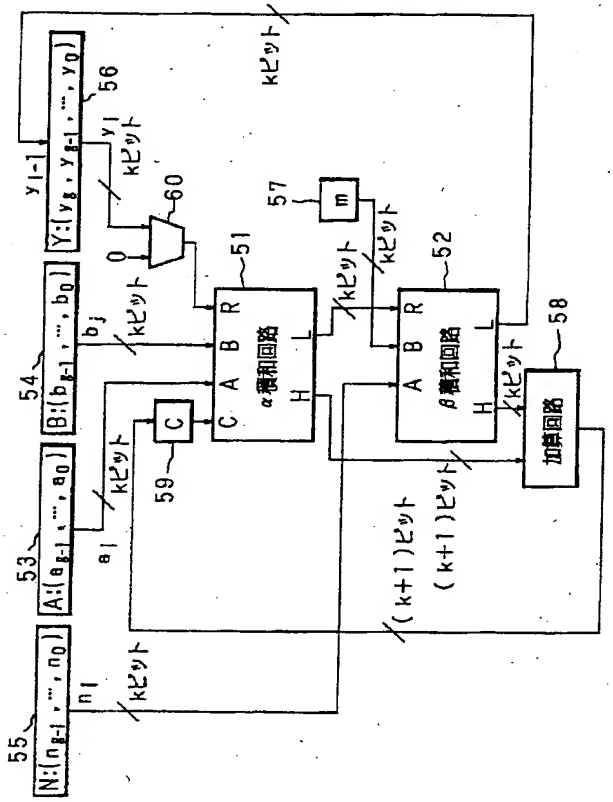
【図８】

コア処理を行うための構成図



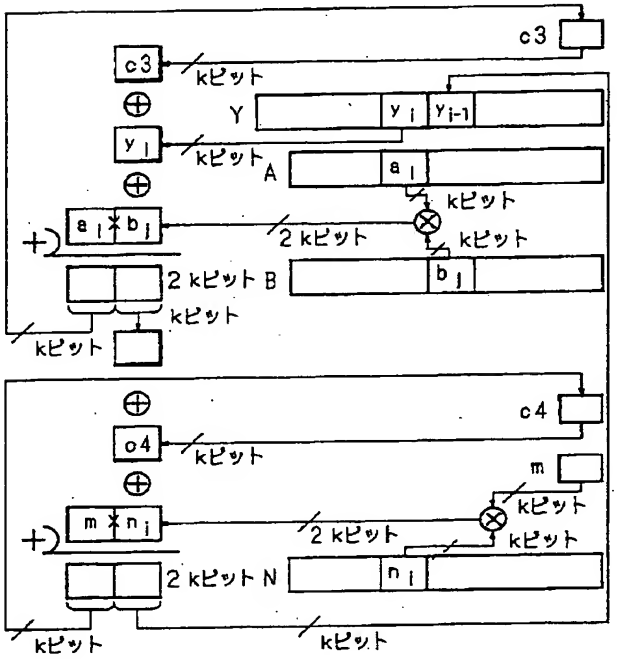
【図１１】

従来の乗算剰余計算装置の構成図



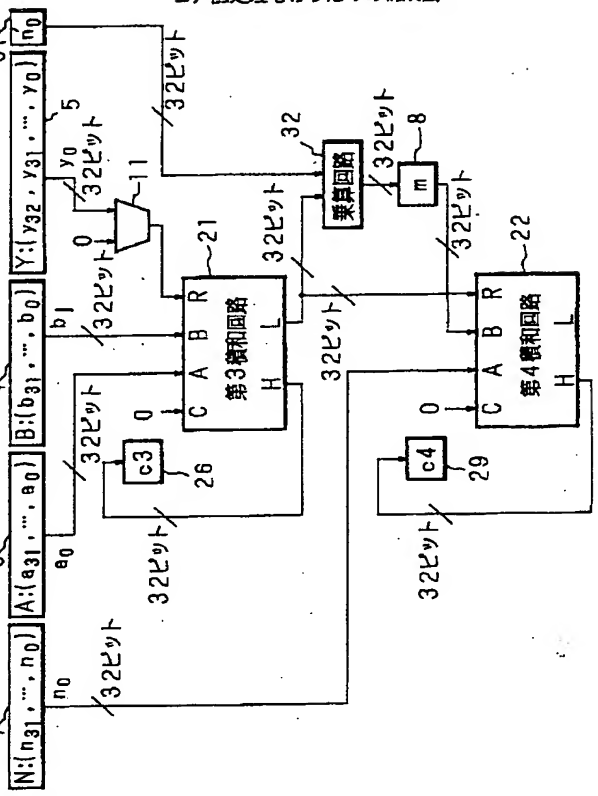
【図５】

本発明の乗算剰余計算装置（第２発明）の動作説明図



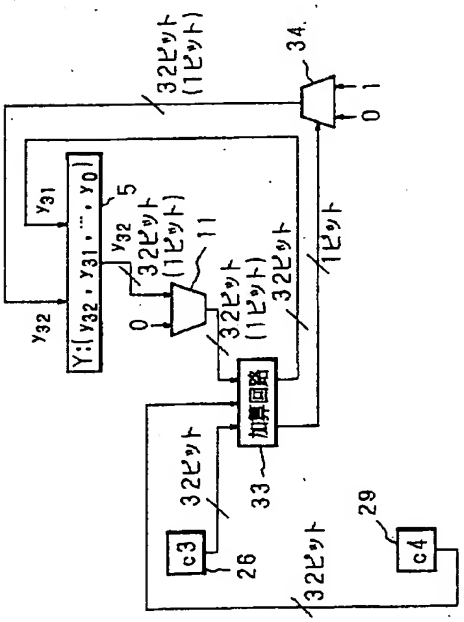
【図７】

コア前処理を行うための構成図



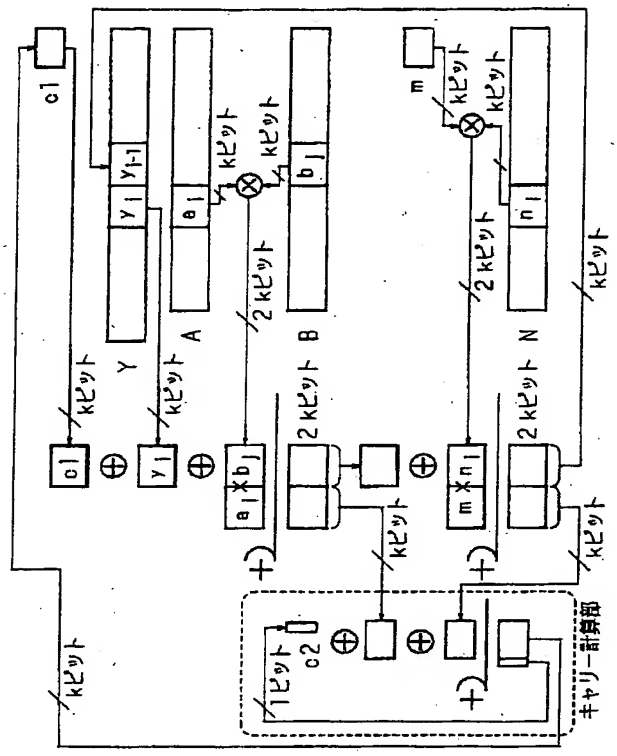
【図９】

コア後処理を行うための構成図



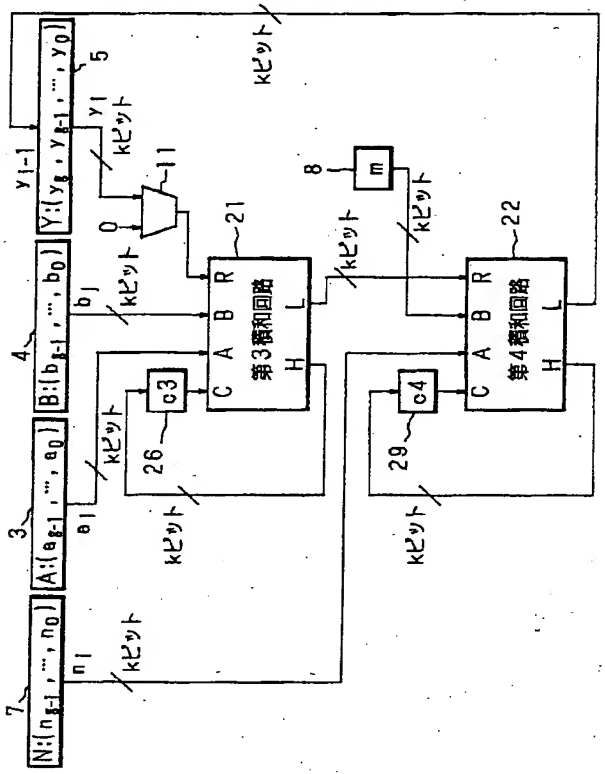
【図３】

本発明の乗算剰余計算装置（第１発明）の動作説明図



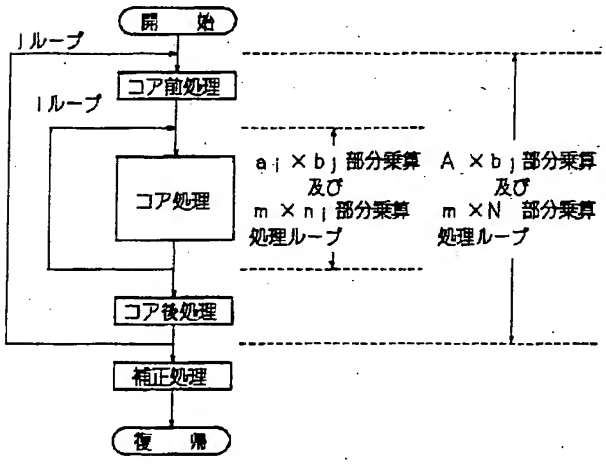
【図４】

本発明の乗算剰余計算装置（第２発明）の構成図



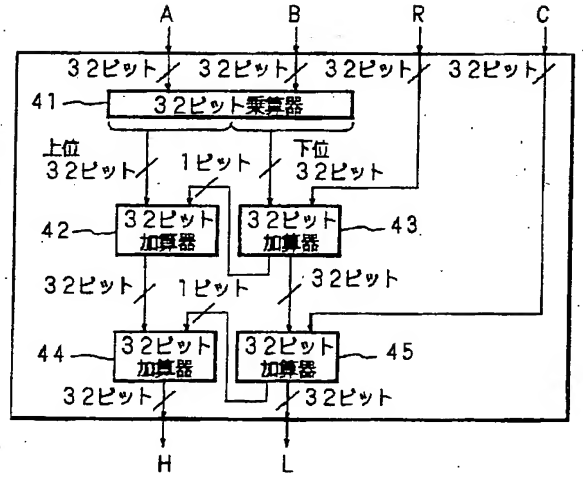
【図６】

モンゴメリ法による乗算剰余処理の一例を示すフローチャート



【図１０】

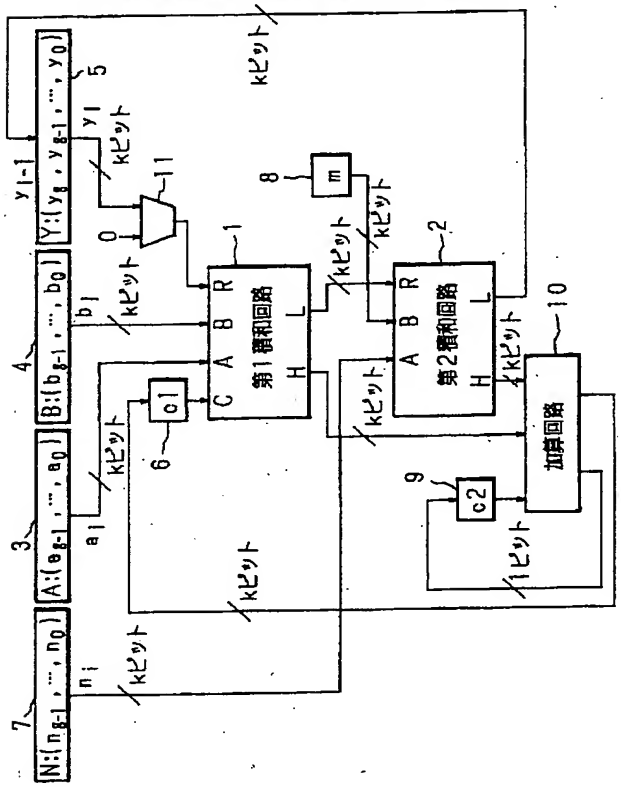
積和回路の構成図



フローチャートである。
【図７】コア前処理を行うための構成図である。
【図８】コア処理を行うための構成図である。
【図９】コア後処理を行うための構成図である。
【図１０】積和回路の構成図である。
【図１１】従来の乗算剰余計算装置の構成図である。
【図１２】従来の積和回路の構成図である。
【図１３】従来の乗算剰余計算装置の動作説明図である。
【符号の説明】
1　第１積和回路
2　第２積和回路
3　Ａレジスタ
4　Ｂレジスタ
5　Ｙレジスタ
6　ｃ1 レジスタ
7　Ｎレジスタ
8　ｍレジスタ
9　ｃ2 レジスタ
10　加算回路
21　第３積和回路
22　第４積和回路
26　ｃ3 レジスタ
29　ｃ4 レジスタ

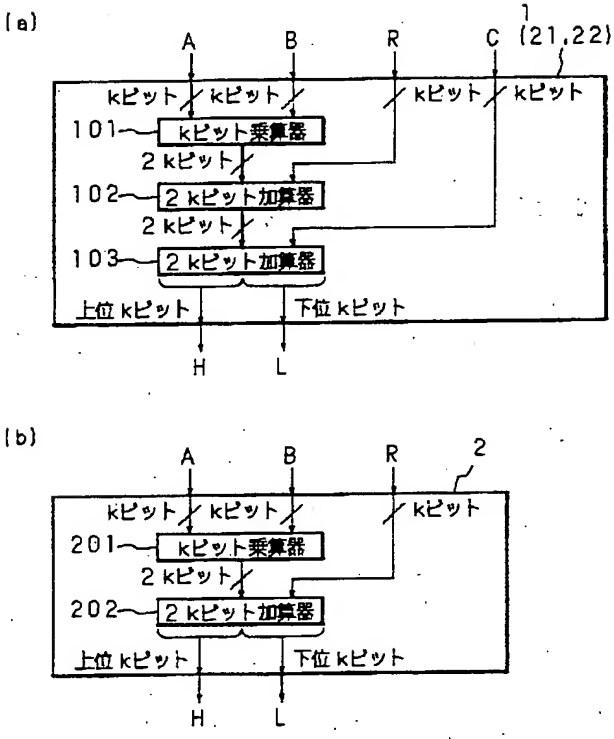
【図１】

本発明の乗算剰余計算装置（第１発明）の構成図



【図２】

積和回路の構成図



ビットをｃ3 レジスタ26に格納し、下位32ビットを第３積和回路21とパラメータｍを計算するための乗算回路32とへ出力する。乗算回路32は、第３積和回路21の出力とレジスタ31の出力ｎ′０とを乗算し、その乗算結果の下位32ビットをｍレジスタへに出力する。

【００７３】第４積和回路22は、Ｎレジスタ７からの入力ｎ0 とｍレジスタ８の値とを乗算し、その乗算結果と第３積和回路21からの出力とを加算する。なお、コア処理と同じ積和回路を使用する場合は、更にその結果と０とを加算する。そして、結果の上位32ビットをｃ4 レジスタ29に格納する。下位32ビットは使用しない。

【００７４】（コア処理）図８は、ｉループ内部処理であるコア処理を行う構成の一例を示す図である。Ｙレジスタ５は前回の処理結果の保持及び今回の処理結果の出力用レジスタである。選択回路11は、（アルゴリズム３）でＹ＝０の処理に相当するものである。

【００７５】第３積和回路21は、まず、Ａレジスタ３，Ｂレジスタ４からの入力ａｉ，ｂｊを乗算し、その乗算結果とＹレジスタ５からの入力ｙｉ とを加算し、更にその加算結果とｃ3 レジスタ26の値とを加算する。そして、結果の上位32ビットをｃ3レジスタ26に格納し、下位32ビットを第４積和回路22へ出力する。

【００７６】第４積和回路22は、まず、Ｎレジスタ７からの入力ｎｊ とｍレジスタ８の値とを乗算し、その乗算結果と第３積和回路21からの出力とを加算し、更にその加算結果とｃ4 レジスタ29の値とを加算する。そして、結果の上位32ビットをｃ4 レジスタ29に格納し、下位32ビットをＹレジスタ５のy_{i-1} に格納する。（アルゴリズム３）のＹ＝Ｙ／ｒの処理は、ｉ回目の計算結果をy_{i-1} に格納することで実現している。

【００７７】（コア後処理）図９は、コア後処理を行う構成の一例を示す図である。33はｃ3 レジスタ26の出力とｃ4 レジスタ29の出力と選択回路11の出力とを加算する加算回路、34は加算回路33からのキャリー出力を０，１と比較し、０であれば０を、１であれば１を、Ｙレジスタ５へ出力する選択回路である。

【００７８】このコア後処理では、コア処理終了後のキャリー変数ｃ３，ｃ4 の値の処理を行っている。ｃ3 レジスタ26，ｃ4 レジスタ29の値及びＹレジスタ５からの入力y_{32}を加算回路33に入力し、その加算結果をＹレジスタ５のy_{31}に出力し、キャリーを処理単位である32ビットの値に変換してＹレジスタ５のy_{32}に出力する。ここで、出力からもわかるように、y_{32}の値はＹレジスタ５では32ビットとして扱われているが、実際は１ビットの値であるので、加算結果は32ビット＋キャリーの範囲で収まる。

【００７９】（積和回路の構成）図10は、上述の構成例で用いた積和回路の構成の一例を示す図である。ここでは、全ての処理単位を32ビットになるように構成している。積和回路は、１個の32ビット乗算器41と、４個の32ビット加算器42，43，44，45とを有する。

【００８０】Ａ，Ｂの入力値は32ビット乗算器41で乗算され、上位32ビットと下位32ビットとの２つで出力される。32ビット加算器43は、32ビット乗算器41の出力下位32ビットと入力Ｒの値とを加算し、その加算結果の出力32ビットを32ビット加算器45へ、キャリーを32ビット加算器42へそれぞれ出力する。32ビット加算器42は、32ビット乗算器41の出力上位32ビットと32ビット加算器43のキャリー出力とを加算し、その加算結果の出力32ビットを32ビット加算器44へ出力する。この加算ではキャリーが発生しないことが理論的に証明されている。

【００８１】32ビット加算器45は、32ビット加算器43の出力と入力Ｃの値とを加算し、その加算結果の出力32ビットは積和回路のＬ出力（下位32ビット）となり、キャリーは32ビット加算器44へ出力される。32ビット加算器44は、32ビット加算器42の出力と32ビット加算器45のキャリー出力とを加算し、その加算結果の出力32ビットは積和回路のＨ出力（上位32ビット）となる。この加算ではキャリーが発生しないことが理論的に証明されている。

【００８２】なお、本発明の乗算剰余計算装置は、ハードウェアに限らず、同様の機能構成の少なくとも一部をソフトウェアで実現することもでき、そのような場合にも処理の高速化を達成することができる。例えば、ソフトウェアで第２発明の乗算剰余計算装置を実現し、パイプライン処理が可能な32ビットプロセッサ上で実行した場合、乗算剰余処理時間を比較すれば、従来の方式の約半分の時間で処理が可能となり、その効果が明らかである。

【００８３】

【発明の効果】以上説明したように、第１，第２発明によれば、モンゴメリ法を用いた乗算剰余計算について、全ての処理を処理単位内で行えるので、ＤＳＰ，マイクロコントローラ等の上のソフトウェアとして実現が容易になる。

【００８４】また、第２発明によれば、キャリー伝搬を各積和毎に制限でき、一方の積和処理中にもう一方の積和処理も可能となるので、パイプライン処理などによって、高速に乗算剰余計算を行うことが可能である。

【図面の簡単な説明】

【図１】本発明の乗算剰余計算装置（第１発明）の構成図である。

【図２】積和回路の構成図である。

【図３】本発明の乗算剰余計算装置（第１発明）の動作説明図である。

【図４】本発明の乗算剰余計算装置（第２発明）の構成図である。

【図５】本発明の乗算剰余計算装置（第２発明）の動作説明図である。

【図６】モンゴメリ法による乗算剰余処理の一例を示す

```
    (c4 , c3 )r =c3 +c4 +yg
    yg-1 =c3
    yg =c4
  next  j
  if  Y≧N  then  Y=Y−N
      Y<N  then  return  Y
```

コア後処理

.....

補正処理

【００６３】図４は、上述したアルゴリズム６のコア処理を実行する乗算剰余計算装置の構成図である。図４に示す乗算剰余計算装置は、内部で乗算及び加算を行う第３積和回路21及び第４積和回路22と、図１に示すものと同様の第１レジスタとしてのＡレジスタ３，第２レジスタとしてのＢレジスタ４，第３レジスタとしてのＹレジスタ５，第５レジスタとしてのＮレジスタ７，第６レジスタとしてのｍレジスタ８及び選択回路11と、キャリー変数ｃ3 を保持する第４レジスタとしてのｃ3レジスタ26と、キャリー変数ｃ4 を保持する第７レジスタとしてのｃ4 レジスタ29とを有する。

【００６４】なお、第３積和回路21及び第４積和回路22の内部構成は、図２（ａ）に示す第１積和回路１の内部構成と同じであり、各積和回路21及び22は、ｋビット乗算器101 と2kビット加算器102 と2kビット加算器103 とから構成されている。

【００６５】第３積和回路21のｋビット乗算器101 は、Ａレジスタ３及びＢレジスタ４からの出力を乗算し、2kビット加算器102 は、ｋビット乗算器101 の出力と選択回路11（Ｙレジスタ５）の出力とを加算し、2kビット加算器103 は、2kビット加算器102 の出力とｃ3 レジスタ26の出力とを加算する。なお、図２（ａ）に示す構成例では、乗算結果に選択回路11（Ｙレジスタ５）の出力を先に加算し、その後にｃ3 レジスタ26の出力を加算するようになっているが、これとは逆に、先にｃ3レジスタ26の出力、その後に選択回路11（Ｙレジスタ５）の出力を加算するように構成しても良い。

【００６６】一方、第４積和回路22のｋビット乗算器101 は、Ｎレジスタ７及びｍレジスタ８からの出力を乗算し、2kビット加算器102 は、ｋビット乗算器101 の出力と第３積和回路21からの下位ｋビットの出力とを加算し、2kビット加算器103 は、2kビット加算器102 の出力とｃ4 レジスタ29の出力とを加算する。なお、図２（ａ）に示す構成例では、乗算結果に第３積和回路21からの下位ｋビットの出力を先に加算し、その後にｃ4 レジスタ29の出力を加算するようになっているが、これとは逆に、先にｃ4 レジスタ29の出力、その後に第３積和回路21からの下位ｋビットの出力を加算するように構成しても良い。

【００６７】図５は、アルゴリズム６のコア処理の内容を示す説明図である。第３積和回路21内にて、Ａレジスタ３の出力ａj （ｋビット）とＢレジスタ４の出力ｂj （ｋビット）とを乗算し、その乗算結果（２ｋビット）に、選択回路11（Ｙレジスタ５）の出力（ｋビット）とｃ3 レジスタ26の出力（ｋビット）とを加算する。なお、選択回路11は、ｊの値と０とを比較し、ｊの値が０である場合には第３積和回路21へ０を出力し、ｊの値が０でない場合には第３Ｙレジスタ５の格納値ｙj を積和回路21へ出力する。第３積和回路21は、その演算結果（２ｋビット）の上位ｋビットをｃ3 レジスタ26へ出力し、その下位ｋビットを第４積和回路22へ出力する。ｃ3 レジスタ26は、このｋビットを次回の演算用のキャリー変数として格納する。

【００６８】第４積和回路22内にて、Ｎレジスタ７の出力ｎj （ｋビット）とｍレジスタ８の出力ｍ（ｋビット）とを乗算し、その乗算結果（２ｋビット）に、第３積和回路21からの下位ｋビット出力を加算する。第４積和回路22は、その演算結果（２ｋビット）の上位ｋビットをｃ4 レジスタ29へ出力し、その下位ｋビットをＹレジスタ５へ出力する。ｃ4 レジスタ29は、このｋビットを次回の演算用のキャリー変数として格納する。また、Ｙレジスタ５は、そのｋビットのデータを値ｙj-1 として格納する。

【００６９】図６は、モンゴメリ法による乗算剰余処理の一例を示すフローチャートである。このフローチャートにおいて、ｊループが（アルゴリズム３）のループ処理に当たる。ｊループの内側では、Ａ×ｂj 及びｍ×Ｎの多重精度×単精度の部分乗算を行っている。ｉループは、Ａ×ｂj 及びｍ×Ｎの多重精度×単精度の計算を単精度×単精度の部分乗算で行っている部分である。ｉループの内部ではａi ×ｂj とｍ×ｎi との部分乗算を行っている。

【００７０】次に、第２発明における更なる具体例について説明する。以下の例では、Ｎ，Ａ，Ｂのビット長を1024ビット、ｇ＝32、処理単位ｋ＝32、Ｒ＝2^{1024}、ｒ＝2^{32}とする。

【００７１】（コア前処理）図７は、コア前処理を行う構成の一例を示す図である。31はモンゴメリ計算用のパラメータｎ′０を保持するレジスタ、32は第３積和回路21の出力とレジスタ31の出力とを乗算する乗算回路である。

【００７２】このコア前処理では、コア処理で使用するｃ3 レジスタ26，ｃ4 レジスタ29及びｍレジスタ８の初期化を行っている。第３積和回路21は、まず、Ａレジスタ３，Ｂレジスタ４からの入力ａ０，ｂj を乗算し、その乗算結果とＹレジスタ５からの入力ｙj とを加算する。なお、コア処理と同じ積和回路を使用する場合は、更にその結果と０とを加算する。そして、結果の上位32

示す乗算剰余計算装置は、内部で乗算及び加算を行う第１積和回路１及び第２積和回路２と、乗算する一方の数Ａ：（a_{g-1}，a_{g-2}，…，a_0）を保持する第１レジスタとしてのＡレジスタ３と、乗算する一方の数Ｂ：（b_{g-1}，b_{g-2}，…，b_0）を保持する第２レジスタとしてのＢレジスタ４と、第２積和回路２の前回の下位ｋビット出力を保持し次回の下位ｋビット出力を格納する第３レジスタとしてのＹレジスタ５と、キャリー変数ｃ1 を保持する第４レジスタとしてのｃ1 レジスタ６と、剰余の法Ｎ：（n_{g-1}，n_{g-2}，…，n_0）を保持する第５レジスタとしてのＮレジスタ７と、モンゴメリアルゴリズムにおけるパラータｍを保持する第６レジスタとしてのｍレジスタ８と、キャリー変数ｃ2 を保持する第７レジスタとしてのｃ2 レジスタ９と、第１積和回路１の上位ｋビット出力，第２積和回路２の上位ｋビット出力及びｃ2 レジスタ９の出力を加算するキャリー計算部としての加算回路10と、ｊの値と０とを比較してその出力を選択する選択回路11とを有する。

【００５５】また、第１積和回路１，第２積和回路２の内部構成を図２（ａ），（ｂ）に夫々示す。第１積和回路１は、ｋビット乗算器101 と2kビット加算器102 と2kビット加算器103 とを有する。ｋビット乗算器101 は、Ａレジスタ３及びＢレジスタ４からの出力を乗算し、2kビット加算器102 は、ｋビット乗算器101 の出力と選択回路11（Ｙレジスタ５）の出力とを加算し、2kビット加算器103 は、2kビット加算器102 の出力とｃ1 レジスタ６の出力とを加算する。なお、図２（ａ）に示す構成例では、乗算結果に選択回路11（Ｙレジスタ５）の出力を先に加算し、その後にｃ1 レジスタ６の出力を加算するようになっているが、これとは逆に、先にｃ1 レジスタ６の出力、その後に選択回路11（Ｙレジスタ５）の出力を加算するように構成しても良い。

【００５６】第２積和回路２は、Ｎレジスタ７及びｍレジスタ８からの出力を乗算するｋビット乗算器201 と、ｋビット乗算器201 の出力及び第１積和回路１からの下位ｋビットの出力を加算する2kビット加算器202 とを有する。

【００５７】図３は、アルゴリズム５のコア処理の内容を示す説明図である。第１積和回路１内にて、Ａレジスタ３の出力a_j（ｋビット）とＢレジスタ４の出力b_j（ｋビット）とを乗算し、その乗算結果（２ｋビット）に、選択回路11（Ｙレジスタ５）の出力（ｋビット）とｃ1 レジスタ６の出力（ｋビット）とを加算する。なお、選択回路11は、ｊの値と０とを比較し、ｊの値が０である場合には第１積和回路１へ０を出力し、ｊの値が０でない場合にはＹレジスタ５の格納値y_jを第１積和回路１へ出力する。第１積和回路１は、その演算結果（２ｋビット）の上位ｋビットを加算回路10へ出力し、その下位ｋビットを第２積和回路２へ出力する。

【００５８】第２積和回路２内にて、Ｎレジスタ７の出力n_j（ｋビット）とｍレジスタ８の出力（ｋビット）とを乗算し、その乗算結果（２ｋビット）に、第１積和回路１からの出力下位ｋビットを加算する。第２積和回路２は、その演算結果（２ｋビット）の上位ｋビットを加算回路10へ出力し、その下位ｋビットをＹレジスタ５へ出力する。Ｙレジスタ５は、そのｋビットのデータを値y_{j-1}として格納する。

【００５９】加算回路10は、第１積和回路１からの出力（ｋビット）と第２積和回路２からの出力（ｋビット）とｃ2 レジスタ９からの出力（１ビット）とを加算する。そして、次回の演算用として、その加算結果（ｋ＋１ビット）の上位１ビットをｃ2 レジスタ９へ、その下位ｋビットをｃ1 レジスタ６へ夫々出力する。各ｃ1 レジスタ６，ｃ2 レジスタ９は、これを格納する。

【００６０】（第２発明）キャリー用のレジスタを各積和回路毎に設けた第２発明について説明する。第２発明では、各積和回路におけるキャリー用のレジスタを各別に設けて自身の積和回路にキャリー変数を戻す構成とする。このアルゴリズム６を以下に示す。

【００６１】（アルゴリズム６）乗算する２数Ａ，Ｂ，パラメータＮ′，出力用変数Ｙが何れもｒ進数で、

$A=(a_{g-1}, a_{g-2}, \cdots, a_0)_r$，

$B=(b_{g-1}, b_{g-2}, \cdots, b_0)_r$，

$N'=(n'_{g-1}, n'_{g-2}, \cdots, n'_0)_r$，

$Y=(y_g, y_{g-1}, \cdots, y_0)_r$，

$R=r^g$，

$r=2^k$

とあらわされ、ｒ進１桁の一時変数tmp1，キャリー変数ｃ3，ｃ4 とする場合、次に示すｉ，ｊの繰り返し処理によりＡＢR^{-1}mod Ｎを単精度×単精度の計算として求めることができる。

【００６２】

```
Y＝０
for   j＝０ to  g－１                          ……
   (c３, tmp1)r＝y０＋aj・bj
  m＝tmp1・n′０(mod r)                    コア前処理
   (c４, tmp1)r ＝tmp1＋m・n０
for   i＝１ to  g－１                          ……
   (c３, tmp1)r ＝yj ＋c３＋aj・bj
   (c４, yj-1)r ＝tmp1＋m・nj              コア処理
next  i                                        ……
```

和手段とを備え、前記第１積和手段の出力を前記第２積和手段の入力とする構成を有し、前記第２積和手段が演算を行う間、前記第１積和手段で次の回の演算を行うように、パイプライン処理すべく構成したことを特徴とする。

【００４６】請求項11に係るモンゴメリ法による乗算剰余計算装置は、請求項10において、前記第１積和手段及び第２積和手段にあって、各自身の上位出力を各自身の次の回のキャリア入力とするようにしたことを特徴とする。

【００４７】請求項12に係るモンゴメリ法による乗算剰余計算装置は、請求項10において、前記第１積和手段及び第２積和手段における演算量が等しいことを特徴とする。

【００４８】請求項13に係るモンゴメリ法による乗算剰余計算装置は、請求項10において、前記第１積和手段及び第２積和手段は、２つのｋビットの数を乗算する手段と、その乗算結果に２つのｋビットの数を加算する手段とを有することを特徴とする。

【００４９】第１発明の乗算剰余計算装置では、従来例において問題となっている（ｋ＋１）ビットのキャリーを上位１ビットと下位ｋビットとに分離し、その下位ｋビットは１つのキャリーとして従来例と同様に１つ目の積和回路に戻し、その上位１ビットはもう１つのキャリーとして、キャリー計算用の加算回路に戻す。この構成をとることにより、１つ目の積和回路の構成の単純化を図れる。

【００５０】第２発明の乗算剰余計算装置では、２つの積和回路から出力される上位データを加算するのではなく、各積和回路におけるキャリー用のレジスタを各別に設けて自身の積和回路にキャリーを戻す構成とすることにより、第１発明と同様に１つ目の積和回路の構成の単純化を図れると共に、更に各積和回路のキャリー処理が閉じているので、２つ目の積和回路の動作中に、１つ目の積和回路の次回の動作が可能となる。

【００５１】

【発明の実施の形態】以下、本発明をその実施の形態を示す図面を参照して具体的に説明する。

（第１発明）１つ目の積和回路へのキャリーをｋビットにした第１発明について説明する。第１発明では、１つ目の積和回路の出力と２つ目の積和回路の出力との加算結果である（ｋ＋１）ビットのキャリーを上位１ビットと下位ｋビットとに分離し、その下位ｋビットはキャリー変数ｃ１として１つ目の積和回路に戻し、その上位１ビットはもう１つのキャリー変数ｃ２として、キャリー計算用の加算回路に戻す。この場合のアルゴリズム５を以下に示す。

【００５２】（アルゴリズム５）乗算する２数Ａ，Ｂ，パラメータＮ′，出力用変数Ｙが何れもｒ進数で、

Ａ＝（ａg-1，ａg-2，…，ａ0）ｒ，

Ｂ＝（ｂg-1，ｂg-2，…，ｂ0）ｒ，

Ｎ′＝（ｎ′g-1，ｎ′g-2，…，ｎ′0）ｒ，

Ｙ＝（ｙg，ｙg-1，…，ｙ0）ｒ，

Ｒ＝ｒg，

ｒ＝2^k

とあらわされ，ｒ進１桁の一時変数tmp1，tmp2，tmp4，キャリー変数ｃ１，ｃ2とする場合、次に示すｉ，ｊの繰り返し処理によりＡＢR^{-1}mod Ｎを単精度×単精度の計算として求めることができる。

【００５３】

```
Y＝0
for   j＝0  to  g－1                          ……
   (tmp2, tmp1)r ＝y0 ＋aj ・bj
   m＝tmp1・n′0 mod r
   (tmp4, tmp1)r ＝tmp1＋m・n0                 コア前処理
   (c2 , c1 )r ＝tmp2＋tmp4
for   i＝1  to  g－1                          ……
   (tmp2, tmp1)r ＝yj ＋c1 ＋aj ・bj
   (tmp4, yj-1 )r ＝tmp1＋m・nj                コア処理
   (c2 , c1 )r ＝tmp4＋tmp2＋c2
next  i                                       ……
   (c2 , c1 )r ＝(c2 , c1 )r ＋yg
   yg-1 ＝c1                                   コア後処理
   yg ＝c2
next  j                                       ……
if  Y≧N  then  Y＝Y－N                        補正処理
    Y＜N  then  return  Y
```

ここで、（ ）ｒは、括弧内のｒ進数１桁の変数を多重精度として扱うことを示している。またキャリー変数ｃ２はｒ進数１桁で表現しているが、内容は１ビットの値である。

【００５４】図１は、上述したアルゴリズム５のコア処理を実行する乗算剰余計算装置の構成図である。図１に

演算結果を上位１ビットと下位ｋビットとに分けて出力する加算回路と、乗算される２数を保持する第１及び第２レジスタと、前記第２積和回路の下位ｋビット出力を保持し、前記第２積和回路のその次の回の下位ｋビット出力を格納する第３レジスタと、前記加算回路の下位ｋビット出力を保持し、前記加算回路のその次の回の下位ｋビット出力を格納する第４レジスタと、剰余の法を保持する第５レジスタと、モンゴメリのアルゴリズムにおけるパラメータの値を保持する第６レジスタと、前記加算回路の上位１ビット出力を保持し、前記加算回路のその次の回の上位１ビット出力を格納する第７レジスタとを備え、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値及び前記第４レジスタに保持された値を加算する演算を行い、前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力を加算する演算を行い、前記加算回路は、前記第１積和回路の上位ｋビット出力と前記第２積和回路の上位ｋビット出力と前記第７レジスタに保持された値とを加算する演算を行うように構成したことを特徴とする。

【００３７】請求項２に係るモンゴメリ法による乗算剰余計算装置は、請求項１において、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値を加算し、その加算結果に前記第４レジスタに保持された値を加算するように構成したことを特徴とする。

【００３８】請求項３に係るモンゴメリ法による乗算剰余計算装置は、請求項１において、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第４レジスタに保持された値を加算し、その加算結果に前記第３レジスタに保持された数の所定ビットの値を加算するように構成したことを特徴とする。

【００３９】請求項４に係るモンゴメリ法による乗算剰余計算装置（第２発明）は、モンゴメリのアルゴリズムを用いて乗算剰余計算を行う装置において、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第１積和回路と、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第２積和回路と、乗算される２数を保持する第１及び第２レジスタと、前記第２積和回路の下位ｋビット出力を保持し、前記第２積和回路のその次の回の下位ｋビット出力を格納する第３レジスタと、前記第１積和回路の上位ｋビット出力を保持し、前記第１積和回路のその次の回の上位ｋビット出力を格納する第４レジスタと、剰余の法を保持する第５レジスタと、モンゴメリのアルゴリズムにおけるパラメータの値を保持する第６レジスタと、前記第２積和回路の上位ｋビット出力を保持し、前記第２積和回路のその次の回の上位ｋビット出力を格納する第７レジスタとを備え、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値及び前記第４レジスタに保持された値を加算する演算を行い、前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力及び前記第７レジスタに保持された値を加算する演算を行うように構成したことを特徴とする。

【００４０】請求項５に係るモンゴメリ法による乗算剰余計算装置は、請求項４において、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値を加算し、その加算結果に前記第４レジスタに保持された値を加算するように構成したことを特徴とする。

【００４１】請求項６に係るモンゴメリ法による乗算剰余計算装置は、請求項４において、前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第４レジスタに保持された値を加算し、その加算結果に前記第３レジスタに保持された数の所定ビットの値を加算するように構成したことを特徴とする。

【００４２】請求項７に係るモンゴメリ法による乗算剰余計算装置は、請求項４において、前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力を加算し、その加算結果に前記第７レジスタに保持された値を加算するように構成したことを特徴とする。

【００４３】請求項８に係るモンゴメリ法による乗算剰余計算装置は、請求項４において、前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第７レジスタに保持された値を加算し、その加算結果に前記第１積和回路の下位ｋビット出力を加算するように構成したことを特徴とする。

【００４４】請求項９に係るモンゴメリ法による乗算剰余計算装置は、請求項４～８の何れかにおいて、前記第２積和回路による演算中に、前記第１積和回路によりその次の回の演算を行うように構成したことを特徴とする。

【００４５】請求項10に係るモンゴメリ法による乗算剰余計算装置は、暗号化／復号化のためのモンゴンメリ法による乗算剰余計算を行う装置において、乗算演算を行う第１積和手段と、モンゴンメリ剰余演算を行う第２積

【００２６】ここで、（　）$_r$は、括弧内のｒ進数１桁の変数を多重精度として扱うことを示している。tmp3, c_1はｒ進数１桁で表現しているが、内容は１ビットの値である。出力用変数Ｙについて、計算に使用する値がy_jのとき、出力がy_{j-1}に格納されるのは、アルゴリズム３におけるＹ＝Ｙ／ｒの機能をこれにより実現しているためである。また、便宜上、外側のループをｊループ、内側のループをｉループと呼び、ｊループの始めからｉループまでをコア前処理、ｉループ内の処理をコア処理、ｉループの終わりからｊループの終わりまでをコア後処理と呼ぶこととする。

【００２７】図11は、上述したアルゴリズム４のコア処理を実行する乗算剰余計算装置の構成図である。図11に示す乗算剰余計算装置は、内部で乗算及び加算を行うα積和回路51及びβ積和回路52と、乗算する一方の数Ａ：（a_{g-1}，a_{g-2}，…，a_0）を保持するＡレジスタ53と、乗算する一方の数Ｂ：（b_{g-1}，b_{g-2}，…，b_0）を保持するＢレジスタ54と、剰余の法Ｎ：（n_{g-1}，n_{g-2}，…，n_0）を保持するＮレジスタ55と、β積和回路52の出力の下位ｋビットを格納するＹレジスタ56と、モンゴメリのパラータｍを保持するｍレジスタ57と、α積和回路51の出力の上位（ｋ＋１）ビット及びβ積和回路52の出力の上位ｋビットを加算するキャリー計算部としての加算回路58と、加算回路58の加算結果を格納するＣレジスタ59と、ｊの値と０とを比較して出力を選択する選択回路60とを有する。

【００２８】また、α積和回路51，β積和回路52の内部構成を図12（ａ），（ｂ）に夫々示す。α積和回路51は、Ａレジスタ53及びＢレジスタ54からの出力を乗算するｋビット乗算器511 と、ｋビット乗算器511 の出力及び選択回路60（Ｙレジスタ56）の出力を加算する2kビット加算器512 と、2kビット加算器512 の出力及びＣレジスタ59の出力を加算する2k＋１ビット加算器513 とを有する。β積和回路52は、Ｎレジスタ55及びｍレジスタ57からの出力を乗算するｋビット乗算器521 と、ｋビット乗算器521 の出力及びα積和回路51からの下位ｋビットの出力を加算する2kビット加算器522 とを有する。

【００２９】図13は、アルゴリズム４のコア処理の内容を示す説明図である。α積和回路51内にて、Ａレジスタ53の出力a_i（ｋビット）とＢレジスタ54の出力b_j（ｋビット）とを乗算し、その乗算結果（２ｋビット）に、選択回路60（Ｙレジスタ56）の出力（ｋビット）とＣレジスタ59の出力（ｋ＋１ビット）とを加算する。なお、選択回路60は、ｊの値と０とを比較し、ｊの値が０である場合にはα積和回路51へ０を出力し、ｊの値が０でない場合にはＹレジスタ56の格納値y_iをα積和回路51へ出力する。α積和回路51は、その演算結果（２ｋ＋１ビット）の上位（ｋ＋1.）ビットを加算回路58へ出力し、その下位ｋビットをβ積和回路52へ出力する。

【００３０】β積和回路52内にて、Ｎレジスタ55の出力n_j（ｋビット）とｍレジスタ57の出力（ｋビット）とを乗算し、その乗算結果（２ｋビット）にα積和回路51からの下位ｋビット出力を加算する。β積和回路52は、その演算結果（２ｋビット）の上位ｋビットを加算回路58へ出力し、その下位ｋビットをＹレジスタ56へ出力する。Ｙレジスタ56は、そのｋビットのデータを値y_{i-1}として格納する。

【００３１】加算回路58は、α積和回路51からの出力（ｋ＋１ビット）とβ積和回路52からの出力（ｋビット）とを加算し、その加算結果（ｋ＋１ビット）をＣレジスタ59へ出力する。Ｃレジスタ59は、これを格納する。

【００３２】　桁上がり

【発明が解決しようとする課題】上述したような従来の乗算剰余計算装置の構成では、２つの乗算を同一の繰り返しループで行えるが、Ｃレジスタ59から出力される次ループへのキャリーが（ｋ＋１）ビットとなり、そのためα積和回路51の出力が（２ｋ＋１）ビットとなってしまっており、１つ目の積和回路（α積和回路51）の構成を単純化できないという問題がある。

【００３３】また、キャリー計算部である加算回路58において、次回演算用のキャリーが２つのα積和回路51及びβ積和回路52での演算結果に基づいて計算されるため、２つ目の積和回路（β積和回路52）での演算が終了しなければ、次回の１つ目の積和回路（α積和回路51）での演算を行うことができず、処理能率が悪いという問題がある。

【００３４】本発明は斯かる事情に鑑みてなされたものであり、１つ目の積和回路への次回演算用のキャリーをｋビットにすることにより、積和回路の構成を単純化することができ、また、これによりＤＳＰ（Digital Signal Processor），マイクロコントローラ等の上のソフトウェアとして実現が容易になるモンゴメリ法による乗算剰余計算装置を提供することを目的とする。

【００３５】本発明の他の目的は、キャリーの伝搬を各積和回路でのループ内に押さえることにより、２つの積和回路での演算を独立して行え、つまり、２つ目の積和回路での積和を行っている間に１つ目の積和回路で次回の積和を行え、これによりパイプライン処理が可能になるモンゴメリ法による乗算剰余計算装置を提供することにある。

【００３６】

【課題を解決するための手段】請求項１に係るモンゴメリ法による乗算剰余計算装置（第１発明）は、モンゴメリのアルゴリズムを用いて乗算剰余計算を行う装置において、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第１積和回路と、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第２積和回路と、加算演算を行いその

る。上記のアルゴリズムにおいて、入力Ｔは０≦Ｔ＜Ｒ・Ｎを満たす値であるが、実際のＲＳＡ演算では、入力Ｔが整数Ａ，Ｂ（０≦Ａ，Ｂ＜Ｎ）の乗算結果であることが多い。その場合、整数Ａ，Ｂの乗算も多重精度整数演算であるため、多重精度拡張ＲＥＤＣと同様の繰り返し計算が行われる。この場合、乗算とＲＥＤＣとを別々に繰り返し計算すると、繰り返し計算制御によるロスが２倍になってしまう。そこで、乗算とＲＥＤＣとを同一の繰り返しループで行えるように拡張したアルゴリズム３を示す。

【００２１】（アルゴリズム３）ＲＥＤＣを多重精度乗算剰余へ拡張したアルゴリズムＲＥＤＣ（Ａ×Ｂ）は次に示すようになる。乗算する２数Ａ，Ｂ，パラメータＮ′，出力用変数Ｙが何れもｒ進数で、

Ａ＝（ａg-1，ａg-2，…，ａ0）ｒ，
Ｂ＝（ｂg-1，ｂg-2，…，ｂ0）ｒ，
Ｎ′＝（ｎ′g-1，ｎ′g-2，…，ｎ′0）ｒ，
Ｙ＝（ｙg，ｙg-1，…，ｙ0）ｒ，
Ｒ＝ｒg，
ｒ＝2k

とあらわされる場合、次に示すｊ＝０～ｇ－１の繰り返し処理により、ＡＢＲ-1mod Ｎを多重精度×単精度の計算として求めることができる。

【００２２】Ｙ＝０
for　ｊ＝０　to　ｇ－１
Ｙ＝Ｙ＋Ａ・ｂj
ｍ＝ｙ0・ｎ′0 mod ｒ
Ｙ＝Ｙ＋ｍ・Ｎ
Ｙ＝Ｙ／ｒ
next
if　Ｙ≧Ｎ　then　Ｙ＝Ｙ－Ｎ
Ｙ＜Ｎ　then　return　Ｙ

このようにして得られるＡＢＲ-1mod Ｎと、上述したように予め求めておいたR2 mod Ｎとの積で再びＲＥＤＣを行うことにより、ＡＢＲ-1mod Ｎを求めることができる。

【００２３】（ＲＥＤＣの単精度×単精度処理への拡張）アルゴリズム３では、多重精度のモンゴメリ乗算剰余を多重精度×単精度で実現可能としているが、この多重精度×単精度の計算部分をさらに単精度×単精度の計算を組み合わせて行えるよう拡張する。この場合、Ａ×ｂj の計算部分とｍ×Ｎの計算部分とが繰り返し計算となり、上述の場合と同様に２つの乗算を別々に繰り返し計算すると、繰り返し計算制御によるロスが２倍になってしまう。そこで、２つの乗算を同一の繰り返しループで行えるようにすれば、ロスの低減が可能である。２つの乗算を同一の繰り返しループで行えるように拡張したアルゴリズム４を示す。

【００２４】（アルゴリズム４）ＲＥＤＣを単精度×単精度へ拡張したアルゴリズムＲＥＤＣ（Ａ×Ｂ）は次に示すようになる。乗算する２数Ａ，Ｂ，パラメータＮ′，出力用変数Ｙ，キャリー変数Ｃが何れもｒ進数で、

Ａ＝（ａg-1，ａg-2，…，ａ0）ｒ，
Ｂ＝（ｂg-1，ｂg-2，…，ｂ0）ｒ，
Ｎ′＝（ｎ′g-1，ｎ′g-2，…，ｎ′0）ｒ，
Ｙ＝（ｙg，ｙg-1，…，ｙ0）ｒ，
Ｃ＝（ｃ1，ｃ0）ｒ，
Ｒ＝ｒg，
ｒ＝2k

とあらわされ、ｒ進１桁の一時変数tmp1，tmp2，tmp3，tmp4とする場合、次に示すｉ，ｊの繰り返し処理によりＡＢＲ-1mod Ｎを単精度×単精度の計算で求めることができる。

【００２５】

```
Ｙ＝０
for　ｊ＝０　to　ｇ－１                                  ……
  (tmp2, tmp1)ｒ＝ｙ0＋ａj・ｂj
  ｍ＝tmp1・ｎ′0 mod ｒ
  (tmp4, tmp1)ｒ＝tmp1＋ｍ・ｎ0                  コア前処理
  (ｃ1，ｃ0)ｒ＝tmp2＋tmp4
for　ｉ＝０　to　ｇ－１                                  ……
  (tmp3, tmp2, tmp1)ｒ＝ｙj＋(ｃ1，ｃ0)ｒ＋ａj・ｂj
  (tmp4, ｙj-1)ｒ＝tmp1＋ｍ・ｎj                  コア処理
  (ｃ1，ｃ0)ｒ＝tmp4＋(tmp3, tmp2)ｒ
next　ｉ                                                 ……
  (ｃ1，ｃ0)ｒ＝(ｃ1，ｃ0)ｒ＋ｙg
  ｙg-1＝ｃ0                                      コア後処理
  ｙg＝ｃ1
next　ｊ                                                 ……
if　Ｙ≧Ｎ　then　Ｙ＝Ｙ－Ｎ                          補正処理
  Ｙ＜Ｎ　then　return　Ｙ
```

ように組み合わせて用いられる場合が多い。

【００１０】公開鍵暗号系の中で、現在最も有力なものが1977年にリヴェスト（Rivest），シャミア（Shamir）及びエイドルマン（Adlman）の三人によって発明されたＲＳＡ暗号である。このＲＳＡ暗号の基本原理は次のようなものである。

【００１１】（ＲＳＡの基本アルゴリズム）暗号鍵（ｅ，Ｎ）と対応する復号鍵（ｄ，Ｎ）とにおいて、ｅとＮとは公開鍵であり、ｄは秘密鍵である。平文をＭ，暗号文をＣとすると、暗号化Ｅと復号Ｄとのアルゴリズムは次のようにあらわされる。

$C = E(M) = M^e \bmod N$

$M = D(C) = C^d \bmod N$

但し、$d \cdot e = 1 \bmod LCM[(p-1),(q-1)]$

$N = p \cdot q$

ＬＣＭ：最小公倍数（lowest common multiple）

ｐ，ｑは大きな素数

【００１２】通常、ｅ，ｄ，Ｍ，Ｎなどは1024ビット程度の大きな整数が用いられているので、高速指数計算法を使用しても１回のＲＳＡ演算で平均1500回程度の多重精度乗算と剰余算とを行わなければならない。特に剰余計算は、近似法，剰余テーブル方式，モンゴメリのアルゴリズム等、多くの高速化手法が提案されている。このような、ＲＳＡ暗号に代表される公開鍵暗号系の多くで利用される、べき乗剰余アルゴリズムを高速に処理するためには、１回あたりの剰余アルゴリズムの高速化が要求される。

【００１３】この剰余演算の高速化を実現する一方法であるモンゴメリのアルゴリズムについて説明する。

（モンゴメリのアルゴリズム）モンゴメリのアルゴリズムは、剰余の法Ｎ（Ｎ＞１）と、剰余の法Ｎと互いに素である基数Ｒ（Ｒ＞Ｎ）とを用いると、被剰余数Ｔから$TR^{-1} \bmod N$の計算が基数Ｒによる除算のみで行えることを利用して、Ｎによる除算を用いることなく剰余計算を行うアルゴリズムである。ここで、Ｎ，Ｎ′，Ｒ，R^{-1}及びＴは整数であり、被剰余数Ｔは$0 \leqq T < R \cdot N$、R^{-1}は剰余の法Ｎの上での基数Ｒの逆数であり、$R \cdot R^{-1} - N \cdot N' = 1$（$0 \leqq R^{-1} < N$，$0 \leqq N' < R$）の関係を満たす。

【００１４】更に、この基数Ｒに２のベキ乗数を使用した場合、基数Ｒによる除算をシフト操作に置き換えることができるため、$T \rightarrow TR^{-1} \bmod N$の計算の高速処理が可能となる。次に、アルゴリズム１として、$T \rightarrow TR^{-1} \bmod N$のアルゴリズムＲＥＤＣ（Ｔ）を示す。但し、アルゴリズム１において（Ｔ＋ｍ・Ｎ）／Ｒは必ず割り切れることが証明されている。

【００１５】（アルゴリズム１）$T \rightarrow TR^{-1} \bmod N$のアルゴリズムＹ＝ＲＥＤＣ（Ｔ）は次のようにあらわされる。

$M = (T \bmod R) \cdot N' \bmod R$

$Y = (T + M \cdot N) / R$

if　$Y \geqq N$　then　$Y = Y - N$

$Y < N$　then return　Ｙ

【００１６】１回のＲＥＤＣでは、剰余Ｔmod Ｎではなく$TR^{-1} \bmod N$が求められるだけである。よって、剰余Ｔmod Ｎを求めるためには、次に示すようにＲＥＤＣ（Ｔ）と予め求めておいた$R^2 \bmod N$との積で、再びＲＥＤＣを行えば良い。

ＲＥＤＣ（ＲＥＤＣ（Ｔ）・（$R^2 \bmod N$））

$= (TR^{-1} \bmod N) \cdot (R^2 \bmod N) \cdot R^{-1} \bmod N$

$= TR^{-1} \cdot R^2 \cdot R^{-1} \bmod N$

$= T \bmod N$

このようにして、剰余Ｔmod Ｎを求めることができる。

【００１７】（ＲＥＤＣの多重精度計算への拡張）次に、剰余の法Ｎまたは基数Ｒが多倍長即ち多重精度である場合について、ＲＥＤＣのアルゴリズムを拡張する。剰余の法Ｎ，基数Ｒが多重精度である場合、ＲＥＤＣの（Ｔmod Ｒ）・Ｎ′及びＭ・Ｎの計算は、多重精度×多重精度の処理となり、汎用の計算機では非常に大きな処理量と処理時間とが必要となる。そこで、この部分を多重精度×単精度の処理で行えるように拡張したアルゴリズム２を示す。

【００１８】（アルゴリズム２）ＲＥＤＣを多重精度へ拡張したアルゴリズムは次に示すようになる。被剰余数Ｔ，パラメータＮ′，出力用変数Ｙが何れもｒ進数で、

$T = (t_{2g-1}, t_{2g-2}, \cdots, t_0)_r$，

$N' = (n'_{g-1}, n'_{g-2}, \cdots, n'_0)_r$，

$Y = (y_g, y_{g-1}, \cdots, y_0)_r$，

$R = r^g$，

$r = 2^k$

とあらわされる場合、次に示すｊ＝０～ｇ－１の繰り返し処理により$TR^{-1} \bmod N$を多重精度×単精度として求めることができる。ここで単精度とはｒ進数１桁のこととし、同じ文字を使用した場合、基本的に大文字を多重精度、小文字を単精度、小文字の添字を多重精度での桁の位置とする。

【００１９】Ｙ＝Ｔ

for　ｊ＝０　to　ｇ－１

$m = y_0 \cdot n'_0 \bmod r$

$Y = Y + m \cdot N$

$Y = Y / r$

next

if　$Y \geqq N$　then　$Y = Y - N$

$Y < N$　then return　Ｙ

このようにして得られる$TR^{-1} \bmod N$と、上述したように予め求めておいた$R^2 \bmod N$との積で再びＲＥＤＣを行うことにより、$TR^{-1} \bmod N$を求めることができる。

【００２０】（ＲＥＤＣの多重精度乗算剰余への拡張）次に、ＲＥＤＣのアルゴリズムを乗算剰余演算に拡張す

【請求項１０】　暗号化／復号化のためのモンゴンメリ法による乗算剰余計算を行う装置において、乗算演算を行う第１積和手段と、モンゴンメリ剰余演算を行う第２積和手段とを備え、前記第１積和手段の出力を前記第２積和手段の入力とする構成を有し、前記第２積和手段が演算を行う間、前記第１積和手段で次の回の演算を行うように、パイプライン処理すべく構成したことを特徴とするモンゴンメリ法による乗算剰余計算装置。

【請求項１１】　前記第１積和手段及び第２積和手段にあって、各自身の上位出力を各自身の次の回のキャリア入力とするようにした請求項１０記載のモンゴンメリ法による乗算剰余計算装置。

【請求項１２】　前記第１積和手段及び第２積和手段における演算量が等しい請求項１０記載のモンゴンメリ法による乗算剰余計算装置。

【請求項１３】　前記第１積和手段及び第２積和手段は、２つのｋビットの数を乗算する手段と、その乗算結果に２つのｋビットの数を加算する手段とを有する請求項１０記載のモンゴンメリ法による乗算剰余計算装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、例えば公開鍵暗号系のＲＳＡ暗号処理において、モンゴメリのアルゴリズム（Modulo Multiplication Without Trial Division, Peter L. Montgomery, Mathematics of Computation, Volume 44, Number 170, April 1985 pp. 519～528 参照）を用いて乗算剰余計算を高速に行う乗算剰余計算装置に関する。

【０００２】

【従来の技術】近年におけるコンピュータネットワークの発達により、データベースを検索する機会、電子メール，電子ニュース等の電子化された情報をネットワークを経由して送受する機会が急速に増加してきている。更に、これらを利用して、オンラインショッピング等のサービスも提供されつつある。しかし、それに伴って、ネットワーク上の電子化されたデータを盗聴する、改竄する、他人になりすましてサービスを無償で受ける等の問題も指摘されている。特に無線を利用したネットワークにおいては、傍受が容易なためにこれらの問題を防止する対策が望まれている。

【０００３】これらの問題に対して暗号技術を応用した暗号化電子メール，利用者認証システムが提案され、種々のネットワークにも導入されつつある。この意味でコンピュータネットワークにおいては暗号化が必須の技術であるといえる。このような暗号技術の中の一つにディジタル署名即ち認証に適した公開鍵暗号方式があるが、暗号化／復号に大量の処理が必要なために高速化が望まれており、様々な高速化アルゴリズムが発表されている。

【０００４】暗号化方式は、大別すると秘密鍵暗号系と公開鍵暗号系とのニつに分類できる。秘密鍵暗号系は、送信者と受信者とが同じ暗号鍵を持つことにより暗号通信を行う方式である。即ち、秘密鍵暗号系では、あるメッセージを秘密の暗号鍵に基づいて暗号化して相手に送り、受け手はこの暗号鍵を用いて暗号文を復号して元のメッセージに戻して情報を入手する。

【０００５】公開鍵暗号系は、送信者が公開されている受信者の公開鍵でメッセージを暗号化して送信し、受信者が自分の秘密鍵でその暗号化メッセージを復号することにより通信を行う方式である。即ち、公開鍵暗号系では、公開鍵は暗号化のための鍵、秘密鍵は公開鍵により暗号化された暗号を復号するための鍵であり、公開鍵で暗号化した暗号は秘密鍵でのみ復号することができる。

【０００６】秘密鍵暗号系では、個人が秘密に保管しなければならない鍵が通信相手の数だけ必要であり、必要な総鍵数はｎ人のネットワークの場合（ｎ－１）／２個である。また、初めて通信する相手に対しては、何らかの方法で秘密鍵の配送が必要であるという欠点がある。この欠点を解消するために、大規模なネットワークでは鍵管理センタを設置し、センタとの間の秘密鍵のみを保管し、暗号通信を行う場合はセンタから送信相手との秘密鍵を得る方法が用いられる。この場合、秘密鍵の総数はｎ個となる。

【０００７】一方、公開鍵暗号系では、個人が秘密に保管する鍵は自分の秘密鍵のみであり、必要な総秘密鍵数もｎ人のネットワークの場合ｎ個である。また、初めて通信する相手に対しては、公開鍵の配送を行えば良く、鍵管理センタを設置して、ユーザの公開鍵をｎ個公開簿に登録し、センタから送信相手の公開鍵を得る方法が用いられる。この場合、センタは公開鍵の改竄を防ぐだけで、秘密に保管する必要がない。但し、公開鍵方式は秘密鍵方式に比べて鍵のビット数が大きいため保管に要するファイルサイズは大きくなる。

【０００８】また、認証の場合、秘密鍵暗号系では、例えば送信するメッセージを秘密鍵で圧縮変換し、送信文に付加して送り、受信側では同様に圧縮変換して比較する方式がとられている。しかし、送受信が同じ鍵であるため、受信者は認証データを偽造することができる。これに対して、公開鍵暗号系では、秘密鍵で暗号化することができるのは本人だけであるという特徴を利用する。送信者はメッセージを圧縮変換して秘密鍵で暗号化し、送信文に付加して送り、受信者は送信者の公開鍵で付加されたデータを復号し、同様に圧縮変換したものと比較する方式がとられている。この場合、受信者は不正ができない。

【０００９】このように認証系では公開鍵暗号系の技術は必要不可欠であるといえる。しかし、公開鍵暗号系には、暗号化／復号に大量の処理が必要であるという大きな欠点があるため、一般には処理が速い秘密鍵暗号系をメッセージの暗号化に、公開鍵暗号系は認証用にという

【特許請求の範囲】

【請求項１】　モンゴメリのアルゴリズムを用いて乗算剰余計算を行う装置において、

積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第１積和回路と、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第２積和回路と、加算演算を行いその演算結果を上位１ビットと下位ｋビットとに分けて出力する加算回路と、乗算される２数を保持する第１及び第２レジスタと、前記第２積和回路の下位ｋビット出力を保持し、前記第２積和回路のその次の回の下位ｋビット出力を格納する第３レジスタと、前記加算回路の下位ｋビット出力を保持し、前記加算回路のその次の回の下位ｋビット出力を格納する第４レジスタと、剰余の法を保持する第５レジスタと、モンゴメリのアルゴリズムにおけるパラメータの値を保持する第６レジスタと、前記加算回路の上位１ビット出力を保持し、前記加算回路のその次の回の上位１ビット出力を格納する第７レジスタとを備え、

前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値及び前記第４レジスタに保持された値を加算する演算を行い、

前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力を加算する演算を行い、

前記加算回路は、前記第１積和回路の上位ｋビット出力と前記第２積和回路の上位ｋビット出力と前記第７レジスタに保持された値とを加算する演算を行うように構成したことを特徴とするモンゴメリ法による乗算剰余計算装置。

【請求項２】　前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値を加算し、その加算結果に前記第４レジスタに保持された値を加算するように構成した請求項１記載のモンゴメリ法による乗算剰余計算装置。

【請求項３】　前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第４レジスタに保持された値を加算し、その加算結果に前記第３レジスタに保持された数の所定ビットの値を加算するように構成した請求項１記載のモンゴメリ法による乗算剰余計算装置。

【請求項４】　モンゴメリのアルゴリズムを用いて乗算剰余計算を行う装置において、

積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第１積和回路と、積和演算を行いその演算結果を上位ｋビットと下位ｋビットとに分けて出力する第２積和回路と、乗算される２数を保持する第１及び第２レジスタと、前記第２積和回路の下位ｋビット出力を保持し、前記第２積和回路のその次の回の下位ｋビット出力を格納する第３レジスタと、前記第１積和回路の上位ｋビット出力を保持し、前記第１積和回路のその次の回の上位ｋビット出力を格納する第４レジスタと、剰余の法を保持する第５レジスタと、モンゴメリのアルゴリズムにおけるパラメータの値を保持する第６レジスタと、前記第２積和回路の上位ｋビット出力を保持し、前記第２積和回路のその次の回の上位ｋビット出力を格納する第７レジスタとを備え、

前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値及び前記第４レジスタに保持された値を加算する演算を行い、

前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力及び前記第７レジスタに保持された値を加算する演算を行うように構成したことを特徴とするモンゴメリ法による乗算剰余計算装置。

【請求項５】　前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第３レジスタに保持された数の所定ビットの値を加算し、その加算結果に前記第４レジスタに保持された値を加算するように構成した請求項４記載のモンゴメリ法による乗算剰余計算装置。

【請求項６】　前記第１積和回路は、前記第１及び第２レジスタに保持された２数の所定ビットの値を乗算し、その乗算結果に前記第４レジスタに保持された値を加算し、その加算結果に前記第３レジスタに保持された数の所定ビットの値を加算するように構成した請求項４記載のモンゴメリ法による乗算剰余計算装置。

【請求項７】　前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第１積和回路の下位ｋビット出力を加算し、その加算結果に前記第７レジスタに保持された値を加算するように構成した請求項４記載のモンゴメリ法による乗算剰余計算装置。

【請求項８】　前記第２積和回路は、前記第５レジスタに保持された数の所定ビットの値と前記第６レジスタに保持された値とを乗算し、その乗算結果に前記第７レジスタに保持された値を加算し、その加算結果に前記第１積和回路の下位ｋビット出力を加算するように構成した請求項４記載のモンゴメリ法による乗算剰余計算装置。

【請求項９】　前記第２積和回路による演算中に、前記第１積和回路によりその次の回の演算を行うように構成した請求項４～８の何れかに記載のモンゴメリ法による乗算剰余計算装置。

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平11－212456

(43)公開日　平成11年(1999)８月６日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０９Ｃ	1/00	６５０	Ｇ０９Ｃ	1/00	６５０Ａ
Ｇ０６Ｆ	7/72		Ｇ０６Ｆ	7/72	
Ｈ０４Ｌ	9/30		Ｈ０４Ｌ	9/00	６６３Ｂ

審査請求　未請求　請求項の数13　ＯＬ　（全 16 頁）

(21)出願番号　特願平10－14681

(22)出願日　平成10年(1998)１月27日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　武仲　正彦
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　伊藤　孝一
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　鳥居　直哉
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　弁理士　河野　登夫

(54)【発明の名称】　モンゴメリ法による乗算剰余計算装置

(57)【要約】

【課題】　積和回路の構成を単純化することができ、また、パイプライン処理が可能になる、モンゴメリのアルゴリズムを用いて乗算剰余計算を高速に行う計算装置を提供する。

【解決手段】　積和回路21は、Ａレジスタ３及びＢレジスタ４の出力を乗算し、その乗算結果にｃ3 レジスタ26の出力及びＹレジスタ５の出力を加算する。積和回路22は、Ｎレジスタ７及びｍレジスタ８の出力を乗算し、その乗算結果にｃ4 レジスタ29の出力及び積和回路21の出力を加算する。２つの積和回路21，22におけるキャリー用のレジスタ26，29を各別に設けて自身の積和回路にキャリーを戻す構成とする。全ての処理を処理単位（ｋビット）内で行う。積和回路22の動作中に、積和回路21の次回の動作が可能である。

本発明の乗算剰余計算装置（第２発明）の構成図